Experimental Study of Router Buffer Sizing

Neda Beheshti  
Stanford University  
nbehesht@stanford.edu

Yashar Ganjali  
University of Toronto  
yganjali@cs.toronto.edu

Monia Ghobadi  
University of Toronto  
monia@cs.toronto.edu

Nick McKeown  
Stanford University  
nickm@stanford.edu

Geoff Salmon  
University of Toronto  
geoff@cs.toronto.edu

ABSTRACT

During the past four years, several papers have proposed rules for sizing buffers in Internet core routers. Appenzeller et al. suggest that a link needs buffer of size \( O(C/\sqrt{N}) \), where \( C \) is the capacity of the link, and \( N \) is the number of flows sharing the link. If correct, buffers could be reduced by 99% in a typical backbone router today without loss in throughput. Enachescu et al., and Raina et al. suggest that buffers can be reduced even further to 20-50 packets if we are willing to sacrifice a fraction of link capacities, and if there is a large ratio between the speed of core and access links. If correct, this is five orders of magnitude reduction in buffer sizes. Each proposal is based on theoretical analysis, and validated using simulation. Given the potential benefits (and the risk of getting it wrong!) it is worth asking if these results hold in real, operational networks. In this paper, we report buffer-sizing experiments performed on real networks - either laboratory networks with commercial routers as well as customized switching and monitoring equipment (UW Madison, Sprint ATL, and University of Toronto), or operational backbone networks (Level 3 Communications backbone network, Internet2, and Stanford). The good news: Subject to the limited scenarios we can create - the buffer sizing results appear to hold. While we are confident that the \( O(C/\sqrt{N}) \) will hold quite generally for backbone routers, the 20-50 rule should be applied with extra caution to make sure network components satisfy the underlying assumptions.

1. MOTIVATION AND INTRODUCTION

Most routers in the backbone of the Internet have a bandwidth-delay product of buffering for each link; i.e. \( B = 2T \times C \), where \( C \) is the link capacity, and \( 2T \) is the effective two-way propagation delay (RTT) of a TCP flow through the bottleneck link [15, 16, 26]. The value is mandated by Internet RFCs [6] and architectural guidelines, and by the network operators.

On the other hand, several recent papers have proposed considerably reducing the buffers in backbone routers [4, 9, 22]. For example, Appenzeller et al. [4] propose that the buffers can be reduced to \( 2T \times C/\sqrt{N} \), where \( N \) is the number of long-lived flows sharing the link. Throughout the paper we will refer to this as the small buffer model. The basic idea follows from the observation that the buffer size is, in part, determined by the sawtooth window size process of the TCP flows. The bigger the sawtooth, the bigger the buffers need to be so as to guarantee 100% throughput. As the number of flows increases, variations in the aggregate window size process (the sum of all the congestion window size processes for each flow) decreases, following the central limit theorem. The result relies on several assumptions: (1) That flows are sufficiently independent of each other to be desynchronized, (2) That the buffer size is dominated by the long-lived flows, and perhaps most importantly (3) That there are no other significant, unmodeled reasons for buffering more packets. If the result is correct, then a backbone link carrying 10,000 long-lived flows could have its buffer size reduced by a factor of 100 without loss in throughput. If, though, the results are wrong, then the consequences of reducing the buffers in a router, or in an operational commercial network, could be quite severe. The problem is, how to decide if the result is correct, without trying it in an operational network? But who would reduce buffers in an operational network, and risk losing customers’ traffic, before knowing if the result is correct?

It is therefore not surprising that, apart from the results we present here, we are not aware of any backbone commercial network in which the buffers have been reduced to anywhere close to \( 2T \times C/\sqrt{N} \). So the first goal of our work was to experiment with small buffers in laboratory and operational networks.

More recently, Enachescu et al. [9] and Raina et al. [22] have proposed reducing buffers much further to \( O(\log W) \approx 20 – 50 \) packets in backbone routers, where \( W \) is the congestion window size. We will refer to this...
as the tiny buffer model. In [9], the authors reach their conclusion by considering the tradeoff between reducing buffers and losing some throughput – assumed to be 10-20%. In other words, when congested, links behave as if they run at 80-90% of their nominal rates. This could be an interesting assumption in networks with abundant link capacity, or in future optical networks where link capacity might be cheaper than buffers. The results depend on the network traffic being non-bursty, which they propose can happen in two ways: (1) If the core links run much faster than the access links (which they do today), then the packets from a source are spread out and bursts are broken, or (2) TCP sources are changed so as to pace the delivery of packets. If the results are correct, and relevant, then a backbone link could reduce its buffers by five orders of magnitude.

Again, it is difficult to validate these results in an operational network, and we are not aware of any other comprehensive laboratory or network experiments to test the \(O(\log W)\) results. So it is the second goal of our work to experiment with tiny buffers in laboratory and operational networks.

In the remainder of this paper we describe a number of laboratory and network experiments that were performed during 2003 to 2008. The laboratory experiments were performed in the WAIL laboratory at University of Wisconsin Madison, at Sprint Advanced Technology Laboratory, and University of Toronto’s Advanced Packet Switch and Networking Laboratory. Experiments were also performed on the following operational networks: Level 3 Communications’ operational backbone network, Internet2 and Stanford University dormitory traffic. We should make clear that our results are necessarily limited: While a laboratory network can use commercial backbone routers and accurate TCP sources, it is not the same as a real operational backbone network with millions of real users. On the other hand, experiments on an operational network are inevitably limited by the ability to control and observe the experiments. Commercial routers don’t offer accurate ways to set the buffer size, and don’t collect real time data on the occupancy of their queues. And real network experiments are not repeatable for different buffer sizes, making apples-with-apples comparisons difficult.

In laboratory experiments, we generate live TCP traffic (ftp, telnet, or http) using a cluster of PCs or commercial traffic generators. We measure the performance from either real-time or statistical traces collected from the system. On one hand, we have a lot of control on the experiments, and can observe almost anything. On the other hand, the traffic is synthetic and might not represent real users. We note that we cannot simply use traces gathered from operational backbone networks for buffer sizing experiments because TCP uses a feedback loop to control congestion, thus live traffic is needed so that we can measure the reaction of flow sources to network conditions.

In our experiments on operational backbone networks we can test the results with real user traffic. However, we have no control over the traffic pattern or the system load. For example, Internet2 has very low load (about 20–30%), which means congestion doesn’t happen naturally. Fortunately, at the time of our experiments, part of the Level 3 Communications network ran at very high link utilization (up to 96%). We report results from both networks. Where possible, we ran experiments over a wide range of operating conditions for both the small buffer and tiny buffer models, including system load ranging from 25% up to 100%, different number of users, various traffic patterns and flow sizes, different propagation delays, access link capacities, and congestion window limits.

The rest of the paper is organized as follows: Section 2 describes the small buffer experiments. We focus on experiments performed on Level 3 Communications operational commercial backbone network, and give a brief overview of other experiments. Section 3 is on tiny buffer size experiments performed on University of Toronto and Sprint ATL. Section 4 concludes the paper.

2. SMALL BUFFERS EXPERIMENTS

We start with, perhaps, the most interesting experiments, which were performed on Level 3 Communications’ operational commercial backbone network. We follow these experiments with a brief overview of other experiments we have conducted in other networks.

2.1 Experiment Setup and Characteristics

Although we have limited control of an operational network, these experiments have several interesting properties. First, the links under study were highly utilized with real live traffic. Their utilization varied between 28.61% and 95.85% during a 24 hour period, and remained above 85% for about four hours every day (a
exceptionally high value - new link capacity was added right after the experiments were completed).

The link under study consisted of three physical, load-balanced links (Figure 1). Traffic at the upstream router is divided equally among the three physical links. Each incoming flow is assigned to one of the three links using a static hash function based on the source-destination IP and port numbers of the flow. Ideally, there is equal traffic on each link (particularly as there are thousands of flows). If we give each physical link a different amount of buffering, we can perform an apples-with-apples comparison between different buffer sizes under almost identical conditions.

The three physical links are OC-48 (2.5Gb/s) backbone links, carrying mostly a mixture of traffic from cable modem and DSL users. Assuming an average rate of 250Kb/s per flow, each link carries about 10,000 flows when highly utilized. The default buffer size was 190ms per-link (60MBytes or 125,000 packets assuming an average packet size of 500B). We reduced the buffer sizes to 10ms (about 3MB or 6,000 packets), 5ms (1.5MB or 3,000 packets), 2.5ms (750KB or 1,500 packets) and 1ms (300KB or 600 packets). Based on the small buffer sizing model, we can expect to need a buffer size of about 2-6ms (depending on the actual value of N).

The buffer sizes were set for 5 days, so they capture the impact of daily and weekly changes in traffic. The whole experiment lasted two weeks in March, 2005. We gathered link throughput and packet drop statistics from each of the three links which were collected by the router every 30 seconds. It would have been preferable to capture all packets and recreate the time series of the buffer occupancy in the router. But the network did not have the facility to do this. Still, we are still able to infer some interesting results.

We also actively injected test flows, measuring the throughput and drops, and compared the performance of the flows going through different links to find out the impact of buffer size reduction. The amount of test flow traffic was kept small.

It is worth noting that the network does not use traffic shaping at the edges or in the core. The routers don’t use RED, and packets are dropped from the tail of the queue.

### 2.2 Experiment Results

During the course of the experiments, we always kept the buffer size on one link at its original size of 190ms, and reduced the buffer size on the other two links. Figure 2 shows the packet drop rate as a function of system load for various buffer sizes. As explained before, both the load and drop rates are measured in time intervals of 30 seconds, and each dot in the graph represents one such interval. Figure 2(a) shows that for buffer sizes of 190ms, 10ms, and 5ms we did not see a single packet drop during the course of the experiments, which lasted more than 10 days for the 190ms buffer size, and about 5 days for each of the 10ms and 5ms buffer sizes!

Reducing the buffers by a factor of forty, without dropping packets, when the utilization frequently exceeds 95% is quite surprising. It suggests that the backbone traffic is very smooth (and presumably any self-similarity has no effect at this degree of multiplexing). Others have also reported smoothness in traffic in core networks [12,13], despite somewhat older results which show self-similarity in core traffic [10,11,23]. Whether this is a result of a shift in traffic pattern and network characteristics over time, or simply the consequence of huge amounts of multiplexing, remains to be determined.

We have found traffic to be extremely smooth in the laboratory and backbone networks we have studied, when there are a large number of flows.

Figure 2(b) shows the drop rate as a function of load, when the buffer sizes are reduced to 2.5ms. This is in the lower part of the range defined by the small buffer sizing model, and as expected we start to observe some packet drops. However, packet drops are still rare; less than 0.02% for the majority of samples. In two samples over five days, we see a packet drop rate between 0.02% and 0.04% and in one sample the drop rate is close to 0.09%.

Figure 2(c) shows what happens when we reduce the buffer size to 1ms. Here, there are a lot more packet drops, which we expect because the buffer size is now about half the value suggested by the small buffer sizing rule. It is interesting to note that almost all the drops occur when the load is above 90% - even though the load value is averaged value over a period of 30 seconds. The instantaneous load is presumably higher, and the link must be the bottleneck for some of the flows. We conclude that having some packet drops does not lead to a reduction in available throughput; it appears that TCP’s congestion control algorithms are functioning well.

Next, we take a closer at link utilizations over time. Figure 3(a) compares the link utilization for the links with 190ms and 1ms of buffering, over three days, but plotting their ratio (i.e. relative utilization) as a function of time. Ideally, the utilization on both links would be equal at all times. However, the differences are not symmetric. The link with 1ms buffering has a slightly higher utilization for the majority of the time.

To further investigate the cause of this asymmetry, we plot link utilizations as a function of time in Figure 3(b). By comparing this graph with Figure 3(a) we...
Figure 2: Packet drop rate as a function of load for different buffer sizes. (a) For buffer sizes equal to 190ms, 10ms, and 5ms, we did not observe any packet drops. (b) For a buffer size of 2.5ms we saw packet drops in only a handful of cases. (c) When the buffer size was set to 1ms we observed packet drops during high utilization time periods.

Figure 3: Comparing the utilization of the links with 1ms and 190ms of buffering. (a) Relative utilization of the two links over time. (b) Individual link utilizations over time. (c) Utilization of 1ms buffer link vs. the utilization of the 190ms buffer link.

observe that during the periods when the overall load of the system is high, the link with 1ms has a slightly higher utilization than the link with 190ms. Figure 3(c) suggests the same, in a different yet more precise way. Each dot in this figure, represents the utilization of the two links in a period of 30 seconds. In this Figure the majority of them fall below the 45 degree line in the graph, which suggests the link with 1 ms of buffering has a higher utilization.

The higher utilization with smaller buffers can be attributed to one of the following two reasons: (1) It might be due to higher loads on the link with 1ms of buffering. Since we have more drops on this link, sources need to send duplicates of the dropped packets, and that might be why we see a higher load. Or, (2) the load balancing scheme might be skewed and might divide the traffic somewhat unevenly among the three links, thus directing more traffic to one of the links.

The question is which of the two reasons is the cause of higher link utilization on one of the two links? The easiest way to answer this question would have been swapping the buffer sizes on the two links under study. Unfortunately, immediately after our experiments, Level 3 upgraded the network and added extra capacity to reduce the load on the links we were studying, which means we could not repeat the experiment with the same conditions.

Interestingly, we see the same phenomena (higher utilization in one link than the others) when the buffer sizes are set to 190ms, and 5ms on two links. Since we did not have any packet drops in these cases, buffer occupancies in both links cannot be affected by packet drops, the RTT of flows must be the same in both links, and therefore (1) cannot be the reason here, i.e. any difference in utilization is most probably not a result of the reaction of TCP sources to packet drops. In other words, we associate these slight differences be-

Based on our experiment results, the load balancing scheme used in this system was changed to one which is believed to be more fair in distributing the load.
between link utilizations with imperfections in the load balancing scheme deployed in the system, rather than the changes in buffer sizes. We conclude that reducing buffer sizes in this network does not have a significant impact on the performance.

2.3 Other Small Buffer Experiments

Other than the experiments on Level 3 Communications’ backbone network, we have also conducted some other experiments on small buffer sizing model. These experiments include University of Wisconsin Madison’s Advanced Internet Laboratory (WAIL), and Stanford University’s dormitory network.

The WAIL experiment, already reported by Appenzeller et al. [4], a cluster of PCs is used to generate up to 400 live TCP flows, and the traffic is directed to a Cisco GSR router. The buffer sizes on the router are reduced by a factor of 10-20, which does not result in any degradation in the system throughput. In Stanford University experiment, we used a Cisco VXR 7200 which connects the dormitories to the Internet via a 100Mb/s link. Traffic is from a mix of different applications including web, ftp, games, peer-to-peer, Streaming and others, and we had 400-1900 flows at any given time. The buffer sizes on the router were reduced by a factor of 20, win no impact on network throughput. We omit the details of these experiments due to limited space, and refer the interested reader to [14] for more details. All these experiments are inline with the small buffers theory, which is based on loose assumptions on the number of flows, their independence. We are fairly confident that the $O(C/\sqrt{N})$ will hold quite generally for backbone routers.

3. TINY BUFFERS EXPERIMENTS

In this section, we describe our experiments on tiny buffer sizing model. These experiments are carried out in the context of the tiny buffer sizing theory: i.e. we consider a single point of congestion, assume core links run much faster than the access links, and expect a 10-20% reduction in network throughput. Without a guarantee that these conditions hold in an operational backbone network, it is not feasible to test tiny buffer model, and therefore, we have to content ourselves to laboratory experiments. We understand this is a limiting factor, and view our work as a first pass in a more comprehensive experimental study of the tiny buffer sizing model by us and others.

3.1 University of Toronto Experiments

We performed an extensive set of experiments on tiny buffer sizing at University of Toronto’s Advanced Packet Switch and Networking Laboratory. During the course of our experiments, we varied several parameters in our network such as buffer size of routers, packet injection times, and hardware level parameters. The goal was to identify the conditions under which tiny buffers are applicable as well as studying the impact of having tiny buffers on network performance. The performance metrics that we studied are link utilization, loss rate, and flow completion times. Link utilization is an important factor from Internet Service Providers (ISPs) point of view while loss rate and flow completion times are end-users’ major concerns. The detailed experiment setup is explained in the following section.

3.1.1 Experiment Setup

Traffic Generation: Generating realistic traffic is one of the key challenges in modeling a network. Experiments in a laboratory setup are often done by using a number of hosts as traffic generators. However, creating a large number of connections, in order to model traffic in networks closer to the core of the Internet, with thousands of flows, is not a trivial task. In our experiments, the traffic is generated using the open-source Harpoon traffic generator [24]. We used a closed-loop version [21] of Harpoon, modified by researchers at the Georgia Institute of Technology. It has been shown in [20] that most of the Internet traffic (60-80%) conforms to a closed-loop flow arrival model. In this model, a given number of users (running at the client hosts) perform successive TCP requests from the servers. The size of each TCP transfer follows a specified random distribution. After each download, the user stays idle for a thinking period that follows another distribution. We also made several further modifications to the closed-loop Harpoon: each TCP connection is immediately closed once the transfer is complete, the thinking period delay is more accurately timed, and client threads with only one TCP socket use blocking instead of non-blocking sockets. For the transfer sizes, we use a Pareto distribution with mean 80KB and shape parameter 1.5. These values are realistic, based on comparisons with actual packet traces [21]. The think periods follow an exponential distribution with mean duration of one second. We performed extensive experiments to evaluate Harpoon’s TCP traffic, the results are provided in the Appendix.

Switching and Routing: One of the major problems we encountered while performing buffer sizing experiments is that commercial routers do not allow high-precision adjustments of their buffer sizes. Moreover, they are not able to provide precise buffer occupancy time-series, which is essential for studying buffer sizing. To address these issues, we used a programmable network component called NetFPGA [2] as the core element of our test-bed. The NetFPGA board is a PCI-form factor board that contains reprogrammable FPGA elements, and four Gigabit Ethernet interfaces, customized for networking experiments. Incoming packets
to a NetFPGA board can be processed, and if needed modified and sent out on any of the four interfaces. We have designed and implemented a NetFPGA-based Ethernet router with finely tunable buffer sizes. One can control the buffer sizes with high precision, based on the number of packets or bytes, and without the worry of hidden buffers in the system.

**Traffic Monitoring:** We have added a module to the NetFPGA-based router to collect accurate buffer occupancy time-series [5], and to accurately measure the bottleneck link’s utilization and loss rate. The system records an event each time a packet is written to, read from or dropped by the router’s output queue. Each event includes the packet size and the precise time it occurred, with an 8 nanosecond granularity. These events are gathered together into event packets, which can be received and analyzed by the computer hosting the NetFPGA board or another computer on the network. The event packets contain enough information to reconstruct the exact queue occupancy over time and to determine the packet loss rate and bottleneck link utilization. This information is vital for experiments involving small packet buffers, which are easily affected by packet timings. To the best of our knowledge, no commercial router today provides these features, and we highly recommend adding such features to future routers.

**Packet Pacing:** Given that backbone networks typically run at 2.5 Gbps or 10 Gbps, and will run even faster in the future, almost every access link runs much slower than the backbone network. For most flows (e.g. where the user is connected to the network via modem, DSL, cable or from a 10Mbps or 100Mbps Ethernet), packets will be naturally spaced out by the network. As packets of a flow cross the boundaries from slower to faster links, the spacing between them should grow.

To study the effect of paced traffic versus bursty traffic, we run two sets of experiments, namely paced and non-paced. In the former set of experiments, the traffic is injected to the network in a paced manner whereas in the latter, the packet arrivals of a flow are more bursty. To emulate slower access links in the paced set of experiments, we used the Precise Software Pacer (PSPacer) package. The PSPacer is installed as a loadable kernel module for the Linux platform and provides precise network bandwidth control and traffic smoothing.

**Packet Delay:** To emulate the long Internet paths in a test-bed it is necessary to add delay to every packet. We used NISTNet [7] delay emulator to introduce propagation delays in the acknowledgement packets that flow from the clients to the servers. The traffic at delay emulator machines is monitored using tcpdump, which captures the headers of every packet. In some circumstances, under high-load tcpdump might not capture a packet, however we observed that the number of such missed packets was negligible: less than 0.1% of the total packets. We used these packet traces to measure the flow completion times and per-flow packet interarrival times.

**Topologies:** In our first topology, illustrated in Figure 4(a), a single point of congestion is formed, where packets from multiple TCP flows go through the NetFPGA router, and share a bottleneck link toward their destinations. Throughout different sets of experiments, we changed the size of the output buffer in the NetFPGA router, and studied how the buffer size affects the utilization and loss rate of the bottleneck link as well as the flow completion times. As mentioned above, the necessity of having smooth traffic is investigated through changing the bandwidth of access links in the network. Our second topology is illustrated in Figure 4(b), where the main traffic (traffic of the bottleneck link) has been mixed with some cross traffic, and separated from it before going through the bottleneck link. The goal is to see if the paced traffic can be changed by the crosscut traffic.

**Host Setup:** In all sets of our experiments, we used the TCP New Reno stack and the maximum advertised TCP window size is set to 20MB, so that transfers are never limited by the window size. We also confirmed that the path MTU is 1500 bytes and that the servers send maximum-sized segments. The aggregated traffic is then carried to the NetFPGA router over the access links, and from the router to the client network over the 1Gbps bottleneck link. The Linux end-hosts and the delay emulator machines are Dell Power Edge 2950 servers running Debian GNU/Linux 4.0r3 (codename Etch) using Intel Pro/1000 Dual-port Gigabit network cards.

### 3.1.2 Experiment Results

In this section, we provide the results of our experimental studies on tiny buffers. We report network performance, including utilization of the bottleneck link, loss rate, and flow completion times under TCP traffic.

To answer the question of how tiny buffers effect the network performance, we use the topology illustrated in Figure 4(a) where a single point of congestion is formed and packets from multiple TCP flows go through the NetFPGA router, and share a bottleneck link toward their destinations. As mentioned in Section 3.1.1, different access network bandwidths are emulated using PSPacer package where users created on a single machine are grouped into classes of different access bandwidths. All users belonging to a class share a single queue with a service rate that is set to 200 Mbps. The size of the queue is chosen to be big enough (5000 packets) so that there is no drop in this buffer. Note that using this approach does not limit a single user’s access rate but emulates low-bandwidth access links to limit
the aggregate data rate of the flows belonging to each class. The slow access links in our setup spread out the bursts in the aggregate ingress traffic of the router, whose input/output links run at 1Gbps. For each experiment, we control the system congestion by fixing the numbers of flows per server. The run time for each experiment is two minutes. To avoid transient effects, we analyze the collected traces after a warm-up period of one minute.

During each experiment, we change the size of the output buffer in the NetFPGA router, and investigate how the buffer size affects the utilization and loss rate of the bottleneck link as well as the flow completion times. To study the impact of tiny buffers on performance, we thus run two sets of experiments namely paced and non-paced in the topology illustrated Figure 4(a). The RTT is fixed to 130 ms in both cases.

**Performance:** Figure 5(a) shows the effect of changing the buffer size on average utilization of the bottleneck link. The total number of flows sharing the bottleneck link is 1200 and 2400. We can see that for tiny buffers (buffer sizes less than 64 packets) paced traffic results in an absolutely higher link utilization than non-paced traffic. For example, with 1200 flows, pacing increased the link utilization from 40% to 64% while having only 4 packet buffers and it increased the utilization from 63% to 82% when we have 2400 flows in the system (The improvement is roughly 20% in both cases). This shows the necessity of pacing requirement as predicted by theory.

Note that in the 1200 flows case, in both paced and non-paced cases the utilization did not grow more than 65%. This is mainly because of the fact that the think time and file size distribution parameters make the number of active users at any time to be roughly half of the total users and most of these active users are setting up many short connections which have very small throughput (almost about 6 packets per RTT) and thus they cannot necessarily saturate the link. Assuming that the average transmission rate is 6 packets per RTT, the total utilization would be roughly 600 (active) × 6 (packets) × 1500 × 8 (MTU) / 0.130 (RTT) which is only about 330 Mbps. The plot shows larger amount (about 650 Mbps) because not all of the flows are this small and hence there are some flows that are sending at a higher rate.

Another interesting observation is that with paced traffic, the link utilization appears to be independent of the buffer size while it is not the case in non-paced traffic. Thus, with paced traffic, there is no need to increase the buffer to achieve a certain link utilization.

Interestingly, we observe that in the 2400 flows case, increasing the buffer size will make the link utilization in the paced traffic to be smaller than non-paced traffic. This has already been observed and reported in [3] and [27]. The main reason is the trade-off between fairness and throughput: In the case when there are many paced flows and the buffer size is big, at the point where the buffer size is about to get full, each flow is sending paced packets at each RTT. These packets get mixed with other flows in a paced manner before getting to the bottleneck router and if the buffer is already full, and number of flows in the network is large, many of flows will experience a loss event and reduce the congestion window to half. Obviously, this is because of the fair behaviour of paced traffic but also will cause the bottleneck link’s utilization to drop. However, with non-paced traffic, since packets arrive at the bottleneck link in large bursts, when the buffer is about to full, a smaller number of unlucky flows will hit the full buffer and experience the loss event and will reduce their rate. Of course, this is unfair as some flows are reducing their rate while other keep on going but it will also keep the link utilization high.

Figure 5(b) compares the loss rate as a function of buffer size for paced and non-paced traffic. We can see that similar to link utilization, for paced traffic the loss rate is almost independent of buffer size whereas it decreases exponentially with non-paced traffic. For tiny buffers, there is a notable reduction in the loss rate for both 1200 and 2400 flow cases. With 1200 flows, the link is not saturated and hence the loss rate for paced traffic is always less than 0.03 percent. However, with tiny buffers, non-paced traffic experienced around 2% drops in average.

**Flow completion Time:** To address the question of
how tiny buffers might affect an individual flow’s completion time, we collected the start and finish times of all the flows going through the bottleneck link, and found the average completion time separately for short and long-lived flows. A short-lived flow is a flow which finishes while in slow start a long-lived flow is a flow which enters congestion avoidance. In our analysis, we considered flows with size smaller than 50 KB (roughly 33 packets) as short-lived flows and flows with size larger than 1000 KB (roughly 600 packets) as long-lived flows. If there is no loss, it takes less than 6 RTTs for the short flows to be completed. Figures 5(c) and 5(d) show the average flow completion times for short-lived and long-lived flows, respectively. We can see that, for 1200 flows case with paced traffic, the flow completion time is independent of the buffer size for both short and long-lived flows whereas in non-paced traffic, increasing the buffer size reduces the flow completion times. In this case, the flow completion time for paced traffic is always smaller than that of the non-paced traffic (for both short and long-lived flows) and there is a notable difference between flow completion times for long-lived flows with tiny buffers.

Cross-cut traffic: To examine the affect of cross-cut traffic on the interarrival time of the main traffic to the bottleneck, we performed a set of experiments with topology illustrated in Figure 4(b) where packets from four several TCP flows go through the bottleneck NetFPGA router toward their destinations. The traffic from Server(1) to Client(2) and from Server(3) to Client(1) is the cross-cut traffic while the traffic from Server(2) to Client(4) and from Server(4) to Client(3) is considered as the main traffic. Figure 7 illustrates the flow of the cross-cut and the main traffic. Throughout different sets of experiments, we change the characteristics of main and cross-cut flows as well as the size of the output buffer in the NetFPGA bottleneck router, and study the effect of cross-cut traffic on the interarrival time of packets to the bottleneck queue.

Figure 6 shows the CDF of interarrival times at the output queue for the bottleneck router in the cross-cut topology, as reported by the NetFPGA. These times only include packets that are actually stored in the queue and not dropped packets. In each experiment there were at most 2400 simultaneous flows. Each figures plots multiple CDFs for experiments using router

![Figure 5:](image)

(a) Link utilization as a function of buffer size for paced and non-paced experiments. (b) Loss rate as a function of buffer size for paced and non-paced experiments. (c) Average flow completion times for short-lived flows. (d) Average flow completion times for long-lived flows.
In network test-beds, where a handful of computers generate traffic representing the communication of hundreds or thousands of actual computers, the configuration of each traffic generator is critically important. Small changes to the software or hardware in a test-bed can have a large impact on the generated traffic and the experiment results, whereas changes to individual machines are unlikely to affect the aggregate traffic at an Internet core router. For an experiment’s results to be relevant to the Internet’s core, the validity of the traffic is paramount.

We have investigated the effects of various software and hardware parameters in the context of buffer sizing experiments, and believe some of these parameters require careful tuning so that the artificially generated traffic mimics the properties of core network flows. For instance, recent network interface cards have many advanced features that can impact the shape of the output traffic, or measurement of various performance metrics. Due to space limitations we describe only two such parameters here, which we believe have the highest impact on our results, and refer the interested reader to [5].

**TCP Segmentation Offload (TSO):** With TSO enabled, the task of chopping big segments of data into packets is done on the network card, rather than in software by the Operating System. The card sends out the group of packets that it receives from the kernel back to back, creating bursty and un-mixed traffic. Clearly, this makes the traffic bursty and highly impacts the results of buffer sizing experiments in a test-bed. Also, TSO must be disabled if packets are being paced in software. When TSO was enabled during our experiments, the gaps between packets added by PSPacer, described in Section 3.1.1, were only added between the large groups of packets sent to the network card, and the resulting traffic on the wire did not have a gap between each packet. Instead, it contained a group of packets back to back followed by a small gap, which was drastically different from the intended traffic pattern.

**Interrupt Coalescing (IC):** To lower the CPU’s interrupt servicing overhead, network cards can coalesce the interrupts caused by multiple events into a single interrupt. With receiver IC enabled, the interarrival time of packets is changed. The network card will delay delivering packets to the operating system while wait-
3.2 Sprint ATL Experiments

Our second set of tiny buffers experiments were conducted in collaboration with Sprint ATL for tiny buffer experiments. Figure 8 shows the topology of the emulated network, which is similar to the setting considered in tiny buffer sizing theory [9]. The core of the experiments is a Juniper T640 router, whose buffers are modified throughout the study. The router is connected to four different networks through four Gigabit Ethernet interfaces. Each cloud in Figure 8 represents one of these networks. The cloud on the left contains all the users/clients, and the three clouds on the right hold the servers. Each server belongs to one of the 99 different subnets (33 for each of the three server networks). The capacity of the access link connecting each server to the rest of the network is set to 15\(\text{Mb/s}\) by default. The requests for file downloads flow from left to right (from clients to servers), and the actual files are sent back from right to left. In this direction, the router has three ingress and one egress line, which means by increasing the load we are able to create congestion on the link connection T640 router to the client network.

In practice, the clients and servers are emulated by two different boxes: Spirent Communications’ Avalanche box plays the role of clients, and the Reflector box plays the role of servers. Each box has four Gigabit Ethernet Interfaces. Obviously, we use only one interface from the Avalanche box, and three interfaces from the Reflector box to connect the boxes to the T640 router. These links correspond to core links, and the link connecting the router to the Avalanche box is the target link. The access links are emulated by the Reflector box, which allows us to change the access link capacity to any desired value. The delay associated with each access link is also emulated by the Reflector box. Since all other link delays are negligible, we can control the two-way propagation delay of packets by modifying these values. In the Appendix, we explain the results of evaluation tests on Avalanche’s TCP traffic.

Throughout the experiments, we use IPMon systems [1] to capture the headers of all the packets which go through the links connecting the router to the Avalanche and Reflector boxes. These headers are recorded along with high precision time-stamps. By matching the packet traces on ingress and egress lines of the router, we can measure the time each packet has spent inside the router, and thus, we can calculate the time-series representing the queue occupancy of the router. This also helps us identify any undocumented buffers inside the router. Such buffers could be fixed-delay buffers (e.g., part of a pipeline, or staging buffers), or could be additional FIFO queues.

3.2.1 Experiment Results

In this section we study the impact of changing buffer sizes on network performance. When allowed by our testing equipment, we also study the effect of changing some other network properties (like traffic patterns, access link properties, number of flows, and others) on buffer sizing results. Due to lack of space, we review some of our results here, and refer the interested reader to [14] for details.

Performance. We reduce the buffer sizes on the router from 8500 packets to just 50 packets, and measure the throughput, drop rate, and delay observed by clients and servers. We found out Avalanche does not enforce rate limitations for incoming traffic, and had to push slow accesses to the server side in this experiment so that we can emulate the impact of slow access links. Avalanche has some other minor timing issues which are described in the Appendix.
individual packets as performance metrics. At 1Gb/s line speed, and an RTT of 50ms, 8500 packets is about twice the bandwidth-delay product, and 50 packets lies in the range of the tiny buffer sizing model.

Figure 9(a) illustrates throughput as a function of time for various buffer sizes, and Figure 9(b) represents the average throughput for different buffer sizes. If we consider the overhead of packet headers, the maximum throughput we can get is about 950Mb/s. We can see that a buffer size between 8500 and 922 packets, gives a throughput of about 100%. This is the range between the rule-of-thumb and the small buffer model. When we push the buffer size to 192, 63, and 50 packets, which is in the range of tiny buffers model, the throughput goes down by 10%, as predicted theoretically. The average level of throughput is maintained very smoothly throughout the experiments, as seen in Figure 9(a).

Figure 9(c) shows that on average packets go through a delay of 155μs to 4.5ms (equivalent to 13 and 375 packets) for buffer sizes between 50 and 8500 packets.

The performance as a function of the number of flows. We would like to see whether the number of flows.

average delays, and the observed reduction in average delay might be a result of activation/deactivation of some hidden buffer inside the router.

For buffer sizes between 922 and 8500 packets, the drop rate is very close to zero (Figure 9(c)). As expected, in these cases utilization is close to 100%. For smaller buffers we see a packet drop rate of up to 0.75%; only 0.25% more than a M/D/1 queue of similar size and arrival rate, confirming once more the smoothness of traffic going through the router.

The impact of increasing network load. In the previous experiment, parameters were chosen so that the effective system load is very close to 100%. What happens if we keep increasing the load? Does the throughput of the network collapse as a result of congestion? This is a valid concern, and to find out the answer, we performed another set of experiments. This time, we varied the potential load of the system between 25% and 150%.

We control the system load by limiting the access link rates, and advertised congestion window, and by changing the number of end users from 150 to 1200.
flows affects the performance of the system. We cannot simply modify the number of flows, since the potential load to the system changes with the number of flows. To fix this problem, we adjust the maximum congestion window size to keep the potential load fixed, when modifying the number of flows in the network. For 150 flows, the maximum congestion window size is set to 64KB. As we increase the number of flows to 300, 600, and 1200, we reduce the maximum congestion window size accordingly (to 32KB, 16KB, and 8KB). The buffer size is set to 85 packets in all these experiments.

Figure 9(f) illustrates the changes in network throughput as we increase the number of flows. When the number of flows is very low (i.e., 150-300) the system throughput is significantly less than 100%. Even when we increase the congestion window size (to increase the potential load), the system throughput is not significantly increased. This can be explained by tiny buffer sizing model as follows: when the number of flows is low, we will not have a natural pacing as a result of multiplexing, and therefore, the throughput will not reach 100%.

Increasing the number of flows beyond a few thousand can result in a significant reduction in throughput, as average the congestion window size becomes very small (2-3 packets or even less), resulting in a very high drop rate and poor performance [8, 17]. This problem is not associated with tiny buffers, and unless we significantly increase the buffer sizes even more than the rule-of-thumb it would not be resolved. We believe this is a result of poor network design and increasing the buffer sizes is not the right way to address such issues.

We have also conducted experiments to study the impact of tiny buffers on performance in the presence of different flow sizes, various access link capacity, and different distributions of RTTs. Our results show performance of a router with tiny buffers does not highly impacted by changes in these parameters. For the sake of space, we omit the details and refer the interested reader to [14].

3.3 Other Tiny Buffer Experiments

Our tiny buffer experiments have verified independently in other test-beds at Lucent Technologies, and Verizon Communications. The only operational network experiment we have done is performed in collaboration with Internet2. We reduced the buffer size of a core router down to 50 packets, while measuring the performance through active flow injections, and passive monitoring tools. Our measurements of throughput and packet drops did not show any degradation. We note that Internet2 operates its network at very low utilization (20-30%); not an ideal setup for buffer sizing experiments. For more details on these experiments, we refer the reader to [14].

Again, all these experiments seem to agree with the tiny buffer sizing rule. Clearly, this rule has more strict assumptions compared to the small buffers rule, and one should be extremely careful to make sure the assumptions hold in an operational network. As discussed in theory, in a network with slow access links the assumptions seem to be satisfied. However, not all backbone traffic comes from slow access links. We conclude that tiny buffer results hold as long as traffic injected to the network is not overly bursty.

4. CONCLUSIONS

The small buffer rule \(O(C/\sqrt{N})\) appears to hold in laboratory and operational backbone networks—subject to the limited number of scenarios we can create. We are sufficiently confident in the \(O(C/\sqrt{N})\) result to conclude that it is probably time, and safe, to reduce buffers in backbone routers, at least for the sake of experimenting more fully in an operational backbone network. The tiny buffer size experiments are also consistent with the theory. We have studied the impact of reducing the buffer size to 20-50 packet under various network conditions. One point that we should emphasize is the importance of the pacing constraint in tiny buffers experiments. As indicated by theory, pacing can happen as a result of slow access links or by modifying sources so as to pace the traffic injected to the network. We find that as long as this constraint is satisfied, we get a good performance. We also have found that some network components (like network interfaces cards) might have features that reduces pacing along the path. Therefore, one should be very careful and aware of such details if tiny buffer sizing result is to be applied in practice.

5. REFERENCES

6. APPENDIX

Harpoon Traffic Generation Evaluation: We have run a large set of experiments to evaluate Harpoon’s TCP traffic. Mainly, we want to verify whether Harpoon’s traffic – which is generated on a limited number of physical machines in our test-bed – can model the traffic coming from a large number of individual TCP sources. If that would be the case, then we should expect to see the same traffic pattern when a fixed number of flows are generated on one single machine, and when the same number of flows are generated on multiple physical machines. In particular we want to know how the flows are intermixed, if a few physical machines are

![Figure 10: Harpoon Evaluation Topology](image-url)
generating them. Our results show that the aggregate traffic becomes less mixed as the number of physical machines becomes smaller.

Figure 10 shows the topology of an experiment run to compare the traffic generated by four machines versus the traffic generated by two machines. In these experiments, a number of connections are created between each pair of physical machines (denoted by the same numbers in the figure). In the first set of experiments, we create a total number of flows (connections) on four pairs of source-destination machines. In the second set, we repeat this experiment by creating the same total number of flows (connections) on four pairs of physical machines (denoted by the same numbers in the figure). The goal is to see how close the traffic of the two experiments look like. In this setup all links run at 1 Gb/s bandwidth, and delay is added by NISTNet to the ack packets. All packets are eventually mixed on a single link (connecting the NetFPGA router to the NISTNet machine). We do our measurements on this shared link.

Figure 11 compares the percentage of successive packets (in the aggregate traffic) which belong to the same flow. The red (darker) bars correspond to the two-source experiment, and the blue bars correspond to the four-source experiment. The plot shows the results for four different settings: Buffer size at the shared link being set to 16 and 350 packets, and maximum TCP window size being set to 64KB, and 20Mb. As can be seen, in all cases packets of individual flows are less likely to appear successively in the aggregate traffic when they are generated by four machines. The difference between the red and the blue bars grows larger as the number of total emulated connections increases. The same result holds when we change the RTT to 50ms and to 150ms.

In most of our buffer sizing experiments with Harpoon generated traffic, we emulate only a few hundred active users on a single machine. This traffic would have been better mixed (and consequently less bursty), had it come from individual TCP sources. Nevertheless, we do not see a negligible difference in the drop rate of packets at the bottleneck link when we change the number of physical machines generating the traffic.

**Avalanche Traffic Generator Evaluation:** In [18] Prasad et al. have shown some discrepancies with Spirent’s TCP implementation. The differences they have observed are between flows generated by Spirent’s traffic generator and those by NewReno TCP with SACK implemented in Linux 2.6.15. The main problems documented in [18] are the followings: Firstly, it has been observed that Spirent’s TCP does not do fast retransmit when the receiver window size is larger than a certain value. Secondly, TCP implementation of Spirent has implemented SACK or NewReno modifications. And finally, the RTO estimation is not consistent and does not seem to confirm to RFC 2988.

We evaluated the Avalanche/Reflector boxes to see if they generate accurate TCP Reno traffic. We started with a single TCP flow, then changed several parameters (link capacity, delay, congestion window size, packet drop rate, etc.) and manually studied the generated traffic. We concluded that packet injection times are accurate, except for a difference of up to 120µs (which can be attributed to processing times and queuing delays in the traffic generators). We also compared the traffic patterns generated by the Avalanche/Reflector boxes with patterns generated by the ns-2 simulator in carefully designed scenarios. While ns-2 is known to have problems of its own, we wanted to identify differences between experimental results and simulation. We found minor differences between ns-2 and Avalanche output traffic. However, we believe these differences do not have any impact on buffer sizing experiments.

---

7These differences were reported to Spirent Communications, and some of them have been resolved in their current systems. We are working with them to resolve the remaining issues.